# EXHIBIT 16

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE NORTHERN DISTRICT OF CALIFORNIA**

| | | |
|---|---|---|
| **IMPLICIT NETWORKS, INC.** | § | |
| | § | |
| **Plaintiff,** | § | |
| | § | |
| | § | |
| | § | |
| | § | |
| **vs.** | § | **Case No. C 10-4234 SI** |
| | § | |
| | § | |
| | § | |
| **JUNIPER NETWORKS, INC** | § | |
| | § | |
| **Defendant.** | § | |

---

**REBUTTAL EXPERT REPORT OF PETER ALEXANDER, PH.D.**

**REGARDING NON-INFRINGEMENT OF**

**U.S. PATENTS NO. 6,629,163 AND 7,711,857**

September 11, 2012

> Plaintiff, in Reply, asserts that the Amendment and Response was simply pointing out that the "novel element" the claim is directed to is the component-by-component rather than overall path storing, and that Implicit was not adding a limitation that the state information could not also relate to the overall sequence or path. Reply at 14. The Court, however, finds Implicit's statement in the reexamination was clear. State information is "not information related to an overall path." This is consistent with the way state information is actually used in the claims and consistent with other language in the claims. See, e.g., '163 Patent Col. 1:48-50 ("retrieving state information relating to performing the processing of the component with the previous packet of the message"); 1:54-56 ("storing state information relating to the processing of the component with packet for use when processing the next packed of the message"); but see Col. 1:39-42 ("storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message"). State information, therefore, is construed as "information specific to a software routine for a specific message that is not information related to an overall path."
>
> See Markman Order at 13:17-14:9. (Emphasis added.)

# 5 THE ACCUSED PRODUCTS

## 5.1 Differences in Juniper Products

71. As indicated above (see paragraph 6), Implicit has accused fourteen distinct Juniper products of infringement in this case. However, Dr. Nettles's report fails to provide any individual analysis of these products on a product-by-product basis. The "elements" section of Dr. Nettles's report (see Appendix A) does not even provide any distinction between different types of SRX products (branch vs. high end) or between SRX and J series products. In fact, Dr. Nettles appears to rely on documents and source code pertaining to products that are not even accused of infringement in this case.

72. It appears that Dr. Nettles improperly assumes that all of the Juniper accused and non-accused products function in the same manner so long as they run some form of the Juniper "JUNOS" operating system. The only support Implicit provides for this conclusion are the conclusory statements of Dr. Nettles and a few citations to Juniper's marketing documents promoting the power of a single operating system.

73. Dr. Nettles's reliance on these marketing documents is misplaced and his conclusion is incorrect. These marketing materials (and Juniper's engineers) explain that JUNOS is a single

operating system only in the sense that there is a "single source code base." JUNOS OS: The Power of One Operating System, at 4. (http://www.juniper.net/us/en/local/pdf/brochures/1500059-en.pdf.); *see also* 6/19/2012 Tavakoli Tr. at 150:30-151:3, Dyckerhoff Tr. at 13-28.

74. A single operating system "does not mean is that all of that software is loaded on each and every system." Tavakoli Tr. at 151:4-19; *see also* 9-20-2012 Narayanaswamy Tr. at 109:5-8. To the contrary, as the JUNOS code base is large and complex, with a variety of pieces that get used for Juniper's different products, so one naturally finds many differences between the products themselves. Dyckerhoff Tr. at 19:3-9 ("But Junos is a very big and complex operating system with many differences between the different products we have, and so, you know, what exactly is shared or not, that's the engineer's job. But if you look at our product portfolio, there's many different things in there, and those work differently inside of the code base."). Dr. Nettles does not even acknowledge that any such differences exist, let alone provide a reasoned explanation for why those differences should not be material to his infringement analysis. Indeed, I would expect just the opposite. For example, the claims are very specific in the type of state information that must be used, as well as the nature of the storing and processing each component must perform for the different types of state information. Thus, even slight variations in product functionality may easily be the difference between infringement and non-infringement in this case.

## 5.2 CPCD Plugin (cpcd_data.c)

75. Dr. Nettles's reliance on the CPCD plugin is one example of the disconnect between his alleged supporting evidence and the Implicit allegations of infringement as to specific Juniper products. Although Dr. Nettles's relies heavily on the source code for the CPCD plugin in an attempt to support his infringement opinions for the accused products, this plugin is not actually even used in the SRX and J series products.

76. Juniper plugins are used in certain products to provide additional, specialized functionality. For example, the CPCD plugin ("Captive Portal Content Delivery") offers the ability to perform IP packet forwarding by re-writing the destination IP address.

77. Juniper uses an approach common in computer programming known as "MAKE" to compile the source code building blocks for its products into executable object code. Each Juniper product package contains the process executable and compiled shared objects that make up a complete process. To do this, Juniper defines a series of files known as "Makefiles." These files may interlate to each other by providing a list of directories containing a dependent Makefile. The MAKE build system processes these dependencies by parsing and then processing the nested Makefiles. Each Makefile processed may also contain a variable holding directory path specification for source files to be compiled and linked in the build.

78. If Dr. Nettles had wanted to demonstrate that the source code he analyzed for the CPCD plugin was actually used in any of the accused products, I would have expected him to provide some evidentiary support or analysis such as an identification of a Makefile or series of Makefiles providing for the cpcd_data.c file to be included in a build for an SRX or J series product. Dr. Nettles does not do this. In fact, other than the single cpcd_data.c file, Dr. Nettles cites to no Juniper technical documentation at all regarding CPCD functionality.

79. I understand that it is Implicit that bears the burden of proving infringement by a preponderance of the evidence. Fundamental to this obligation is the necessity of providing some basis to believe that the source code and other evidence allegedly supporting Dr. Nettles's infringement analysis actually pertain to the accused Juniper products. Having failed to satisfy this basic threshold requirement, Dr. Nettles's opinions regarding infringement necessary fail.

#### 7.1.3.4 The "session_ignore" Error Codes Cannot Satisfy the Limitation of "dynamically identifying a non-predefined sequence of components"

172. Even if a CPCD plugin or other plugin identified in a predefined service set were to be skipped due to an error condition, this would not change the sequence of the other plugins (if any) in that service set.[5] Indeed, Dr. Nettles has not identified any source code that could cause this sequential order to change after the first packet in response to a session ignore message or any other reason. Nor have I seen any source code in my review that could perform this function. As Implicit stated in the reexamination proceedings, dynamically creating the processing path involves both determining which components are necessary and in what order they should be applied. In the case of the Service Set functionality, the order is fixed in advanced and not changed by anything that happens after the first packet. Moreover, the Dr. Nettles theory of removing plugins from a predefined list is in fact the logical opposite of dynamically selecting individual components, in the sense that the latter adds and the former subtracts.

#### 7.1.3.5 Dr. Nettles Fails To Identify the "first packet of the message"

173. In his report on infringement, Dr. Nettles has failed to identify a "first packet" and thus failed to show that the accused Juniper instrumentalities perform all of the requisite steps "for the first packet of the message." I have also looked at other materials and statements from Implicit provided during the discovery phase of this case on this point, including Implicit's responses and supplemental responses to Juniper Interrogatory No. 20, as well as certain briefing to the Court regarding this interrogatory (and the Court's order). Specifically, Implicit stated that: "the first packet is the packet that creates a message specific, stateful data processing path as per the claims. Depending on device configuration, this can be the first control packet, used in establishing the

[5] I have reviewed the technical materials cited in Dr. Nettles's report, including manuals and other papers taken from the Juniper web site, and I have not seen anything suggesting the ability to change the order of plugins to be applied to a message after the first packet of that message is received.

(3) deposition testimony of the parties and third parties; (4) Implicit' responses to Juniper's discovery requests; (5) the documents referenced in Dr. Nettles's report and any other materials relied upon by any of Implicit's experts; (6) publicly available materials.

302. I may also rely on visual aids and demonstrative exhibits that may be prepared based on these materials that illustrate the bases of my opinions, including the materials mentioned in this report and in **Attachment A**. These visual aids and demonstrative exhibits may include, for example, claims charts, patent drawings, excerpts from patent specifications, file histories, deposition testimony, or diagrams or other graphical presentations describing the technology of the patents-in-suit.

303. I also expect to testify at trial with respect to the subject matter of my report and matters addressed by any expert testifying on behalf of Implicit if asked about these matters by the Court or by the parties' counsel. I may also testify on other matters relevant to this case if asked by the Court or by the parties' counsel.

304. To the extent that Implicit or its expert witnesses offer new opinions or evidence in this case, I reserve my right to supplement, amend, or modify the opinions set forth in this report.

Dated: September 11, 2012

Peter Alexander, Ph.D.